Self-Supervised Learning for Molecular Property Prediction

Laurent Dillard
Elix, Inc.
Tokyo, Japan
laurent.dillard@elix-inc.com

Shinya Yuki
Elix, Inc.
Tokyo, Japan
shinya.yuki@elix-inc.com

Abstract

Predicting molecular properties remains a challenging task with numerous potential applications, notably in drug discovery. Recently, the development of deep learning, combined with rising amounts of data, has provided powerful tools to build predictive models. Since molecules can be converted to graphs, Graph Neural Networks (GNNs) have emerged as a popular choice of architecture to tackle this task. Training GNNs to predict molecular properties however faces the challenge of collecting annotations which is a costly and time consuming process. On the other hand, it is easy to access large databases of molecules without annotations. In this setting, self-supervised learning appears as a promising direction to efficiently leverage large sources of unlabeled data and improve the performance of predictive models. In this work, we introduce a self-supervised framework for GNNs tailored specifically for molecular property prediction. Our framework uses multiple pretext tasks focusing on different scales of molecules (atoms, fragments and entire molecules). We evaluate our method on a representative set of GNN architectures and datasets and also consider the impact of the choice of input features. Our results show that our framework can successfully improve performance compared to training from scratch, especially in low data regimes. The improvement varies depending on the dataset, model architecture and, importantly, on the choice of input feature representation.

1 Introduction

Computational methods have been increasingly used in drug discovery pipelines to reduce the number of experiments to carry out to generate new drugs. Predicting molecular properties is critically important to allow virtual screening of large datasets of molecules and select promising candidates to consider for further validation. Although in silico methods such as density functional theory or molecular dynamics simulations allow calculating properties of interest, their computational cost is prohibitively large to apply them in early stages of the drug discovery process where massive numbers of drug candidates need to be explored. Traditional machine learning and, more recently, deep learning methods have therefore been used to allow efficient computation of molecular properties. Specifically, in light of their success in processing graph structured data, Graph Neural Networks (GNNs) have become a popular approach to process molecules by converting them to graphs where atoms and bonds are represented as nodes and edges of the graph.

Like most deep learning applications, training GNNs requires large amounts of annotated data and is therefore faced with the challenge of data scarcity. Especially, for molecular properties, collecting annotations is a costly process as it involves running chemical experiments. On the flip side, many large datasets of molecules have been made publicly available [13, 5, 16] such that it is easy to access large quantities of molecules without annotations. In this context, self-supervised learning has been demonstrated to be effective [18, 15, 12, 23, 29]. One drawback of recent methods however is that

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

they rely on pretext tasks that focus mostly on the atom level only. This is in contrast with molecular properties that are defined on the molecule level. Additionally, although the choice of input features can vary drastically from one method to another [18, 12], the impact of this choice, as far as we are aware, has not been examined in previous studies. Based on those observations, we introduce in this work a new self-supervised learning framework for GNNs tailored specifically for molecular property prediction and analyze how the choice of input features impacts its performance.

2 Related Work

2.1 Molecular Property Prediction

Traditional approaches to molecular property prediction involve the use of Extended Connectivity Fingerprints (ECFP) [17] combined with machine learning models like support vector machines [10], random forest [1] or multi-layer perceptrons [8]. Another line of work makes use of deep architectures like Recurrent Neural Networks (RNNs) [11] and Transformers [21] applied directly to SMILES strings [24]. More recently, graph representations coupled with GNNs have become a popular approach and achieved state-of-the-art performance in many applications [28, 26, 15].

2.2 Self-supervised learning

Self-supervised learning aims at leveraging large quantities of unlabeled data to learn general feature representation using pretext tasks. The pretext tasks should depend on labels that are easy to generate and push the network to learn meaningful features. After self-supervision, the learned representations can be transferred to downstream tasks. This transfer learning setting proves effective especially in cases where the number of samples available for the downstream tasks is limited. Following its recent successes in computer vision [3, 31, 9], self-supervised learning has been increasingly applied to graph structured data and molecular property prediction [23, 29, 30, 32, 18, 12, 20].

3 Method

Devising a self-supervised framework implies designing useful pretext tasks. For molecular properties, many, like ADME properties [2], largely depend on global molecular level characteristics therefore should benefit from pretext task associated with graph level representations. For some molecular properties, such as toxicity, the property can be related to the presence or not of certain specific functional groups and is best understood on the molecular fragment level: it is neither spread across the entire molecule nor is it related specifically to a single atom. Training on the node level is also important as the final graph representation is obtained from the nodes (for reference, Appendix A.1 includes a brief summary about GNNs). Generating meaningful node level representation is a necessary condition to obtain useful graph level representations.
To best align with the goal of predicting molecular properties, our pretext tasks are designed to learn rich feature representation of molecules on 3 distinct scales: atom, fragment and molecule. Our framework is summarized on Figure 1.

3.1 Atom level

A useful pretext task should push the network to learn a good feature representation about the atom context. We defined the atom level pretext task as a classification problem to recognize which fragment each atom belongs to. We detail how the list of fragments to use as labels was generated in Appendix A.2. The loss to be minimized for this task is detailed in Appendix A.3.

3.2 Fragment level

For the fragment level pretext task, the goal is to decompose each molecules into fragments and teach the network to determine which fragments originate from the same molecule. This is cast as a binary classification problem between pairs of fragments from the same molecule (positives) or from distinct molecules (negatives). To obtain fragment representations, the molecular graphs are first partitioned into subsets of nodes, each subset corresponding to a fragment, then edges between distinct subsets are removed. To allow for more diversity, fragment sizes are uniformly distributed

between a minimum and maximum size. The fragment feature vectors are obtained as the average of all node vectors belonging to the fragment.
For a given pair of fragments, the output is obtained by taking the sigmoid of the dot product between the fragments feature vectors. Considering that the batch size is usually relatively large (128, 256, 512...) the problem is very imbalanced with much more negatives than positives. To overcome that issue, we force equal proportions at each batch by considering all positives and randomly sampling an equal number of negatives. The loss to be minimized for this task can be found in Appendix A.3.

3.3 Molecule level

For the molecule level task, we want to push the network to learn feature representations that encode global properties of the graph. To achieve this, we use a multi-label classification task where we use the same fragments introduced for the atom level task 3.1 and predict for each molecule which fragments it contains. The loss to be minimized for this task can be found in Appendix A.3.

The three losses are then combined such that the final loss minimized is:

$$\mathcal{L}_{final} = \lambda_{atom} * \mathcal{L}_{atom} + \lambda_{fragment} * \mathcal{L}_{fragment} + \lambda_{molecule} * \mathcal{L}_{molecule} \quad (1)$$

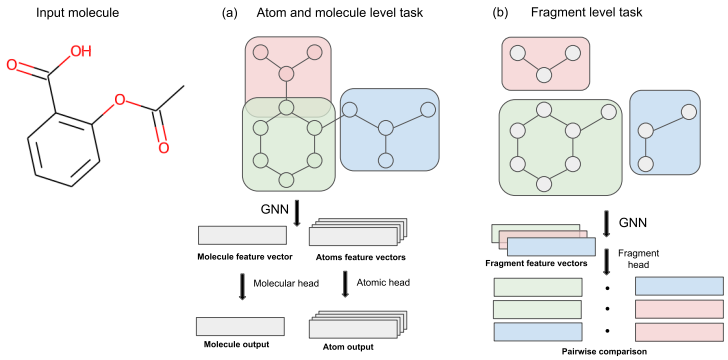

Figure 1: Illustration of our framework. **(a)** For the atom and molecule level tasks, the molecular graph is processed to return atom and molecule features each passed to parallel heads to provide classification outputs. The molecule task is a multi-label classification problem (one molecule can contain multiple fragments) while the atom task is multi-class one (one atom assigned to only one fragment). **(b)** For the fragment task, the molecular graph is decomposed into sub-graphs and processed by the GNN and a fragment head to produce fragment features that are then compared to each other via dot product.

4 Experiments

We evaluate our framework on several standard molecular property prediction benchmark datasets using different popular GNN architectures. We used a subset of ZINC database [13] containing 250,000 samples as the pre-training dataset [7]. For the target datasets, we used a subset of the MoleculeNet [25] benchmark datasets. Further details about the target datasets are provided in Appendix A.4

To ensure that our framework generalizes to multiple GNN architectures, we selected 3 representative GNN architectures: Graph Convolution Network (GCN) [14], Graph Isomorphism Network (GIN) [27] and Directed Message Passing Neural Network (DMPNN) [28], further details about the model architectures and training hyper parameters are provided in Appendix A.5.

On the target datasets, we used scaffold splitting to generate the train, validation and test sets containing respectively 80%, 10% and 10% of the data. Scaffold splitting ensures that different

scaffolds appear in the training, validation and test split and better reflects real world applications [18, 15]. We compared two different settings: training from randomly initialized weights or fine tuning the weights obtained from self-supervision.

Additionally, to measure the impact of the choice of input features, we repeated the experiments using a reduced set of input features. We report the detailed breakdown of the full and reduced set of input features used in Appendix A.6. We also include an ablation study to measure the contribution of each pretext task to our framework in Appendix A.7.

4.1 Results

Dataset	**BACE (1522)**	**BBBP (2053)**	**ClinTox (1491)**	**SIDER (1427)**	**Toxcast (8615)**	**Tox21 (8014)**	**Average**	**Average gain**
GCN	0.831(0.027)	0.898(0.033)	0.909(0.038)	0.605(0.027)	0.651(0.014)	0.769(0.013)	0.777(0.025)	
GCN (SSL)	0.858(0.023)	0.904(0.035)	0.927(0.026)	0.615(0.017)	0.653(0.012)	0.761(0.024)	0.786(0.023)	+0.009
GIN	0.844(0.028)	0.885(0.038)	0.902(0.045)	0.602(0.023)	0.625(0.013)	0.773(0.012)	0.772(0.027)	
GIN (SSL)	0.854(0.025)	0.891(0.032)	0.904(0.033)	0.614(0.017)	0.630(0.013)	0.773(0.018)	0.778(0.023)	+0.006
DMPNN	0.800(0.034)	0.894(0.038)	0.908(0.029)	0.620(0.021)	0.637(0.012)	0.762(0.018)	0.770(0.025)	
DMPNN (SSL)	0.855(0.027)	0.898(0.034)	0.910(0.040)	0.605(0.025)	0.618(0.013)	0.757(0.020)	0.774(0.026)	+0.004
GCN †	0.717(0.048)	0.864(0.038)	0.630(0.121)	0.572(0.023)	0.624(0.009)	0.715(0.048)	0.687(0.042)	
GCN (SSL) †	0.856(0.019)	0.896(0.037)	0.687(0.051)	0.592(0.019)	0.649(0.004)	0.755(0.017)	0.739(0.025)	+ 0.052
GIN †	0.816(0.038)	0.893(0.032)	0.560(0.070)	0.576(0.023)	0.598(0.019)	0.740(0.022)	0.697(0.033)	
GIN (SSL) †	0.849(0.032)	0.904(0.026)	0.605(0.148)	0.594(0.029)	0.623(0.011)	0.750(0.015)	0.721(0.043)	+0.014
DMPNN †	0.676(0.037)	0.833(0.051)	0.509(0.108)	0.564(0.024)	0.603(0.018)	0.710(0.039)	0.649(0.046)	
DMPNN (SSL) †	0.831(0.032)	0.877(0.041)	0.595(0.109)	0.594(0.014)	0.598(0.033)	0.733(0.016)	0.704(0.041)	+0.055
GROVER [18]	**0.894(0.028)**	**0.940(0.019)**	**0.944(0.021)**	**0.658(0.023)**	**0.737(0.010)**	**0.831(0.025)**	**0.834(0.021)**	
GROVER-GIN [18]	0.862(0.020)	0.925(0.036)		0.648(0.015)				
Hu et al.	0.851(0.027)	0.915(0.040)	0.762(0.058)	0.614(0.006)	0.714(0.019)	0.811(0.015)	0.778(0.028)	

Table 1: Evaluation of our self-supervised framework denoted by (SSL). For each architecture the baseline results correspond to training from randomly initialized weights. Two SOTA methods have been included for comparison. †denotes experiments where the reduced set of input features were used. Shaded results indicate best between baseline and SSL while **bold** results indicates best performance overall. For GROVER, GROVER-GIN and Hu et al., results were taken from [18].

For each experiment, we measured the ROC-AUC on the test set and report both the mean value and standard deviation across 10 runs. The results obtained are presented in Table 1. When using the full set of input features, we observe that the impact of our self-supervised learning framework largely depends both on the dataset and model architecture. For the same dataset, improvement can vary from +0.01 to +0.055 ROC-AUC depending on the model used. Conversely, for the same model the impact of self-supervised learning can vary from +0.055 to -0.019 ROC-AUC depending on the dataset. When using the reduced set of input features, the benefits of self-supervision increased substantially: from an average of +0.009 to +0.052 for GCN, +0.006 to +0.014 for GIN and +0.004 to +0.055 for DMPNN. We interpret these results the following way: if a lot of information the network can learn from self-supervision is already contained in the input features then the performance gain is marginal. On the contrary, if a more limited set of input features is used, self-supervision is more beneficial as it can more easily improve upon the initial feature representation.

Comparing with other state-of-the-art (SOTA) methods, GROVER [18] largely outperforms all other experiments. GROVER uses self-supervision combined with a new architecture, based on transformers and GNNs. Its performance significantly decreases when using using a standard GNN architecture like GIN. However, GROVER-GIN still outperforms all other methods on the reported datasets. It should be noted that GROVER and GROVER-GIN performed self-supervised learning on a dataset of 11 millions molecules while we only used 250k. Using our framework on a larger pre-training set might further improve the performance but we leave these experiments for future work.
Compared to Hu et al., our framework performs comparably while again using less data for the pre-training phase (250k vs 2 millions). Additionally our framework does not require pre-processing the entire pre-training dataset contrary to GROVER nor does it require an auxiliary GNN to be trained contrary to Hu et al..

5 Conclusion

In this work, we applied self-supervised learning methods for GNNs specifically in the context of molecular property prediction. We introduced a framework that uses a combination of 3 tasks focusing each on a different scale of molecules: atom, fragment and entire molecules. We evaluated our framework on 6 datasets from MoleculeNet using scaffold splits and 3 different GNN architectures. For each architecture, we compared our framework to a baseline of training the network from randomly initialized weights. Finally we analyzed the impact of the choice of input features on our framework.

Our results indicate that self-supervision can successfully improve the performance of GNNs for molecular property prediction, especially in low data regime. However, our framework was not able to improve the performance consistently across datasets and architectures. Another important finding highlighted in this work is the importance of the choice of input features for self-supervision. When using a very limited set of input features, the gain in performance obtained by applying our self-supervised framework increased significantly and was consistent across all datasets and GNN architectures tested. This finding is consistent with the results of previous studies, where GROVER [18], that used a rich set of input features, reported a lower improvement on their baseline than Hu et al. [12], that used a much more reduced set.

References

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL http://dx.doi.org/10.1023/A%3A1010933404324.

[2] Darko Butina, Matthew D Segall, and Katrina Frankcombe. Predicting adme properties in silico: methods and models. *Drug Discovery Today*, 7(11):S83–S88, 2002. ISSN 1359-6446. doi: https://doi.org/10.1016/S1359-6446(02)02288-2. URL https://www.sciencedirect.com/science/article/pii/S1359644602022882.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020. URL https://arxiv.org/abs/2002.05709.

[4] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *J. Cheminformatics*, 1:8, 2009. URL http://dblp.uni-trier.de/db/journals/jcheminf/jcheminf1.html#ErtlS09.

[5] Anna Gaulton, Anne Hersey, Michał Nowotka, A. Patrícia Bento, Jon Chambers, David Mendez, Prudence Mutowo, Francis Atkinson, Louisa J. Bellis, Elena Cibrián-Uhalte, Mark Davies, Nathan Dedman, Anneli Karlsson, María Paula Magariños, John P. Overington, George Papadatos, Ines Smit, and Andrew R. Leach. The ChEMBL database in 2017. *Nucleic Acids Research*, 45(D1):D945–D954, 11 2016. ISSN 0305-1048. doi: 10.1093/nar/gkw1074. URL https://doi.org/10.1093/nar/gkw1074.

[6] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212, 2017. URL http://arxiv.org/abs/1704.01212.

[7] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, Feb 2018. ISSN 2374-7943. doi: 10.1021/acscentsci.7b00572. URL https://doi.org/10.1021/acscentsci.7b00572.

[8] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[9] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *CoRR*, abs/1911.05722, 2019. URL http://arxiv.org/abs/1911.05722.

[10] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998. doi: 10.1109/5254.708428.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997.

[12] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Pre-training graph neural networks. *CoRR*, abs/1905.12265, 2019. URL http://arxiv.org/abs/1905.12265.

[13] Shoichet Brian K Irwin John J. Zinc–a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 2005. doi: 10.1021/ci049714. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1360656/.

[14] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907.

[15] Pengyong Li, Jun Wang, Yixuan Qiao, Hao Chen, Yihuan Yu, Xiaojun Yao, Peng Gao, Guotong Xie, and Sen Song. Learn molecular representations from large-scale unlabeled molecules for drug discovery. *CoRR*, abs/2012.11175, 2020. URL https://arxiv.org/abs/2012.11175.

[16] PubChem. PubChem. URL https://pubchem.ncbi.nlm.nih.gov/.

[17] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, May 2010. ISSN 1549-9596. doi: 10.1021/ci100050t. URL https://doi.org/10.1021/ci100050t.

[18] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. 2020.

[19] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017. URL http://arxiv.org/abs/1708.07120.

[20] Fan-Yun Sun, Jordan Hoffmann, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *CoRR*, abs/1908.01000, 2019. URL http://arxiv.org/abs/1908.01000.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[22] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets, 2016.

[23] Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. Molclr: Molecular contrastive learning of representations via graph neural networks. *CoRR*, abs/2102.10056, 2021. URL https://arxiv.org/abs/2102.10056.

[24] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.*, 28(1):31–36, February 1988. ISSN 0095-2338. doi: 10.1021/ci00057a005. URL https://doi.org/10.1021/ci00057a005.

[25] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay S. Pande. Moleculenet: A benchmark for molecular machine learning. *CoRR*, abs/1703.00564, 2017. URL http://arxiv.org/abs/1703.00564.

[26] Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of Medicinal Chemistry*, 63(16):8749–8760, Aug 2020. ISSN 0022-2623. doi: 10.1021/acs.jmedchem.9b00959. URL https://doi.org/10.1021/acs.jmedchem.9b00959.

[27] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *CoRR*, abs/1810.00826, 2018. URL http://arxiv.org/abs/1810.00826.

[28] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, Andrew Palmer, Volker Settels, Tommi Jaakkola, Klavs Jensen, and Regina Barzilay. Analyzing learned molecular representations for property prediction. *Journal of Chemical Information and Modeling*, 59(8):3370–3388, Aug 2019. ISSN 1549-9596. doi: 10.1021/acs.jcim.9b00237. URL https://doi.org/10.1021/acs.jcim.9b00237.

[29] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *CoRR*, abs/2010.13902, 2020. URL https://arxiv.org/abs/2010.13902.

[30] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. *CoRR*, abs/2106.07594, 2021. URL https://arxiv.org/abs/2106.07594.

[31] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *CoRR*, abs/2103.03230, 2021. URL https://arxiv.org/abs/2103.03230.

[32] Shichang Zhang, Ziniu Hu, Arjun Subramonian, and Yizhou Sun. Motif-driven contrastive learning of graph representations. *CoRR*, abs/2012.12533, 2020. URL https://arxiv.org/abs/2012.12533.

[33] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar Tatikonda, Nicha C. Dvornek, Xenophon Papademetris, and James S. Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *CoRR*, abs/2010.07468, 2020. URL https://arxiv.org/abs/2010.07468.

A Appendix

A.1 Preliminaries: GNN architectures

GNNs can be simply described using the message passing framework [6]. Each node in a graph sends and receives message from its neighbors. An update function is used to update each node vector based on the messages it received.

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote an undirected graph with $\mathbf{X}$ the input feature matrix encoding the node vectors and $\mathbf{E}$ the input feature matrix encoding the edges vectors, $\mathcal{N}(v)$ denoting the neighborhood of node v and n denoting the layer index. The GNN message passing is formulated as follows:

$$h_v^0 = \mathbf{X}_v \tag{2}$$

$$m_v^{n+1} = \sum_{w \in \mathcal{N}(v)} M_n(h_v^n, h_w^n, E_{vw}) \tag{3}$$

$$h_v^{n+1} = U_n(h_v^n, m_v^{n+1}) \tag{4}$$

The choice of of message and update function at each layer M_n, U_n depends on the GNN architecture. After N layers, a permutation invariant readout function R can be applied to obtain a single feature vector for the whole graph.

$$g(\mathcal{G}) = R(\{h_v^N | v \in \mathcal{G}\}) \tag{5}$$

This graph representation can then be processed by a multi-layer perceptron (MLP) to generate the desired output.

A.2 Fragments used for atom and molecule level tasks

To generate the labels used for the molecule and atom level pretext tasks, we used the list of fragments introduced for Synthetic Accessibility Score [4] (SAScore) computation as our available classes. This list of fragments is based on a structural analysis of roughly 1 million representative molecules from the PubChem database [16]. Each fragment is associated with its frequency of appearance. Considering that the total number of resulting fragments is 605,864 and that most fragments are

extremely rare (51% appear only once), we restricted ourselves to only the top 2000 most frequent fragments which covers fragments that appear at least 1000 times in the database. This leads to a classification problem with 2000 classes. For each atom, we consider whether it belongs to any of the 2000 fragments. If it does, we assign its label to be the largest fragment it belongs to. If it belongs to none of the fragment then no loss will be computed on this atom during training. Considering that we use the most frequent fragments, it is rare that an atom is not assigned any label. On our pre-training dataset, this only happened for 1.16% of all atoms. We also considered using more fragments, training on 4000 classes but empirically found it had no impact on the framework. This is consistent with the fact that additional classes only represent very rare fragments from which little useful knowledge can be gained.

A.3 Pre-training tasks

A.3.1 Atom level

The loss to be minimized for this task for each batch can be written as follows:

$$\mathcal{L}_{atom} = \frac{1}{|A|} \sum_{h_v^N, y_v \in A} \mathcal{L}(y_v, f_a(h_v^N)) \tag{6}$$

Where A denotes the set of atoms in the batch (excluding those that were not assigned any fragment), h_v^N denotes the feature representation of node v obtained following equations (2) to (4), y_v denotes the fragment label associated to node v, $\mathcal{L}$ denotes the cross entropy loss and f_a denotes a MLP head used to obtain the atom classification output.

A.3.2 Fragment level

The loss to be minimized for this task for each batch can be written as follows:

$$x_{\mathcal{F}} = \frac{1}{|\mathcal{F}|} \sum_{v \in \mathcal{F}} h_v^N \tag{7}$$

$$\mathcal{L}_{fragment} = \frac{1}{|N| + |P|} \Big(\sum_{\substack{\mathcal{F}, \mathcal{F}' \in P \\ \overline{\mathcal{F}}, \overline{\mathcal{F}'} \in N}} \mathcal{L}(1, \sigma(f_p(x_{\mathcal{F}}) \cdot f_p(x_{\mathcal{F}'}))) + \mathcal{L}(0, \sigma(f_p(x_{\overline{\mathcal{F}}}) \cdot f_p(x_{\overline{\mathcal{F}'}}))) \tag{8}$$

Where $\mathcal{F}$ denotes both a fragment and the set of nodes belonging to it, h_v^N denotes the feature representation of node v. P and N denote respectively the set of positive and selected negative pairs in the batch, σ denotes the sigmoïd activation function, f_p denotes a projection head, $\cdot$ denotes the dot product operation and $\mathcal{L}$ denotes the binary cross entropy loss.

A.3.3 Molecule level

The loss to be minimized for this task for each batch can be written as follows:

$$\mathcal{L}_{molecule} = \frac{1}{|B|} \sum_{\mathcal{G}, y \in B} \mathcal{L}(y, f(g(\mathcal{G}))) \tag{9}$$

Where B denotes the batch, g denotes the GNN function detailed in equation (5), f denotes a MLP head to generate the multi-label classification output and $\mathcal{L}$ denotes the binary cross entropy loss function.

A.4 Target datasets

For the target datasets, we used a subset of the MoleculeNet [25] benchmark datasets. Namely we used the following 6 classification datasets:

- **BACE**: This dataset contains 1,522 samples with binary labels on binding results with human β-secretase 1 (BACE-1).

- **BBBP**: This dataset contains 2,053 samples with binary labels about permeability of the molecule with the blood-brain barrier. Blood-brain barrier penetration is relevant for the design of drugs impacting central nervous system.
- **ClinTox**: This dataset contains 1,491 samples with two distinct binary labels associated. The first label refers to clinical trial toxicity and the second one to the FDA approval status.
- **SIDER**: This dataset contains 1,427 samples with binary labels for 27 different drug side-effects categories.
- **ToxCast**: This dataset contains 8,615 samples with 617 binary labels based on the results of in vitro toxicology experiments.
- **Tox21**: This dataset contains 8,014 samples with 12 binary labels based on toxicity measurements.

MoleculeNet contains more datasets but we chose to focus mainly on datasets with a lower number of samples since this better reflects practical applications of self-supervised learning for drug discovery.

A.5 Model training

For GCN, we used 3 graph layers with a feature dimension of 512 and concatenation of max and sum as readout function. For GIN we used 5 graph layers with a feature dimension of 512 and concatenation of max and sum as readout function. For DMPNN we used a single graph layer with 4 steps with a feature dimension of 512 and set2set [22] as readout function. All architectures used a 2 layers MLP head on top of the obtained graph feature representations to produce the final output.

During pre-training, we used the loss defined in equation (1) with λ_{atom}, $\lambda_{fragment}$, $\lambda_{molecule}$ all set to 1. Each model was trained for 40 epochs with a batch size of 512 using AdaBelief optimizer [33] with OneCycle learning rate policy [19] with a learning rate and weight decay of 1e-4 and 1e-5 respectively.
On the target datasets, each model was trained for 40 epochs with a batch size of 128 using AdaBelief and OneCycle policy. The learning rate and weight decay were set to 1e-3 and 1e-5 for all target datasets except BACE and BBBP which used a learning rate of 3e-4 and 5e-4 respectively.

A.6 Input features

Two different sets of input features were used for the experiments: a full and a reduced one. Each set is detailed in the following Tables 2, 3.

Category	Property	Dimension
atom	Encoding of atom type	1
(total dimension = 2)	Encoding of atom chirality	1
bond	Enconding of bond type	1
(total dimension = 2)	Encoding of bond direction	1

Table 2: Summary of all properties encoded in the atom and bond features used as input features for the reduced set set.

A.7 Ablation study

We conducted an ablation study to investigate the relative contribution of each of the 3 tasks of our self-supervised learning framework. To that end, we ran 3 different experiments where each model was trained on only one of the 3 tasks. Then we ran an additional experiment selecting the two tasks that performed best individually and compared all results to the complete framework that uses all 3 tasks. The results are presented in Table 4.

Here again the performance depends largely on the dataset with some task performing well individually on certain datasets (molecule only on BACE dataset) while leading to negative transfer learning on other datasets (molecule only on Toxcast and Tox21). On average, the atom level task reaches

Category	Property	Dimension
atom (total dimension = 74)	One hot encoding of atom type	43
	One hot encoding of atom degree	11
	One hot encoding of atom valence	7
	Atom formal charge	1
	Number of radical electrons	1
	One hot encoding of atom hybridization	5
	Binary encoding is atom aromatic	1
	One hot encoding of total number of hydrogens	5
bond (total dimension = 12)	One hot encoding of bond type	4
	Binary encoding is bond conjugated	1
	Binary encoding is bond in ring	1
	One hot encoding of bond stereochemistry	6

Table 3: Summary of all properties encoded in the atom and bond features used as input features for the full set.

the best individual performance even though it is outperfomed by the other two on half the datasets (BACE, BBBP and Tox21). The combination of atom and fragment level tasks reaches a better average performance than any of the two individually and adding the third task brings and additional improvement in average performance. These results suggest that fine tuning the λ_{atom}, $\lambda_{fragment}$, $\lambda_{molecule}$ parameters depending on the dataset should be beneficial to reach optimal performance but using the same value of 1 for all gives a good baseline.

Dataset	**BACE**	**BBBP**	**ClinTox**	**SIDER**	**Toxcast**	**Tox21**	**Average**
GCN	0.831(0.027)	0.898(0.033)	0.909(0.038)	0.605(0.027)	0.651(0.014)	**0.769(0.013)**	0.777(0.025)
GCN (SSL)	0.858(0.023)	0.904(0.035)	**0.927(0.026)**	0.615(0.017)	**0.653(0.012)**	0.761(0.024)	**0.786(0.023)**
GCN SSL atom only	0.848(0.019)	0.905(0.033)	0.894(0.042)	0.614(0.024)	0.651(0.008)	0.766(0.016)	0.780(0.024)
GCN SSL fragment only	0.837(0.028)	**0.908(0.039)**	0.900(0.040)	0.611(0.028)	0.651(0.010)	**0.769(0.014)**	0.779(0.027)
GCN SSL molecule only	**0.861(0.028)**	0.898(0.037)	0.905(0.041)	0.610(0.024)	0.635(0.008)	0.752(0.023)	0.777(0.027)
GCN SSL atom + fragment	0.850(0.024)	0.904(0.037)	0.899(0.040)	**0.621(0.022)**	0.652(0.010)	**0.769(0.020)**	0.783(0.025)

Table 4: Comparing the impact of each task of the self-supervised learning framework. 4 distinct settings are considered: using only the atom level task, using only the fragment level task, using only the molecule level task and using the combination of atom and fragment level tasks. Best performance overall is indicated in **bold**.